# AN OVERVIEW OF THE PHILIPS DIALOG SYSTEM

*Harald Aust*[*], *Olaf Schröer*[**]

[*]Philips Speech Processing, Weisshausstr. 2, 52066 Aachen, Germany, aust@pfa.research.philips.com
[**]Philips Speech Processing, Kackertstr. 10, 52072 Aachen, Germany, schroer@acn.be.philips.com

## ABSTRACT

In 1994, we introduced the world's first publicly available natural-language dialog system, which has given information on the schedule of the German railways. Since then, the software package used for the implementation of this system has been improved in various ways.

In addition, the original train timetable information system has not remained the sole application: A number of other systems have become available during the past few years and are now in everyday use.

In this paper, we will give a brief overview of our system, the underlying technology, and the most important applications realized so far.

## 1. INTRODUCTION

Automatic inquiry systems, as we normally call this particular subset of telephone-based human-machine dialog systems, are systems that people can call in order to obtain certain information, or to have a transaction performed, without a human operator being involved. In their most basic form, they consist of a hierarchy of menus, each of which offers a number of alternative actions to choose from. Callers can navigate up and down the structure by using the keys of their telephones or, in some cases, simple speech commands.

Especially in the USA, these systems have become widely used. They are based on relatively reliable and cheap technology, but are, on the other hand, not very user-friendly. What we really would like to have are systems with which a caller can communicate with natural, unrestricted, and fluent speech, in the same way he or she would talk to a human operator, without having to follow a rigid hierarchy of menus. In other words: The system should adapt to the caller, not vice versa.

While many research groups world-wide are working on these natural-language dialog systems, and even though a number of prototypes have become available during the last couple of years, some of which accessible also by people outside the respective organizations that created them, it seems that the Philips system is still the only one that actually has left the laboratory and is in everyday use.

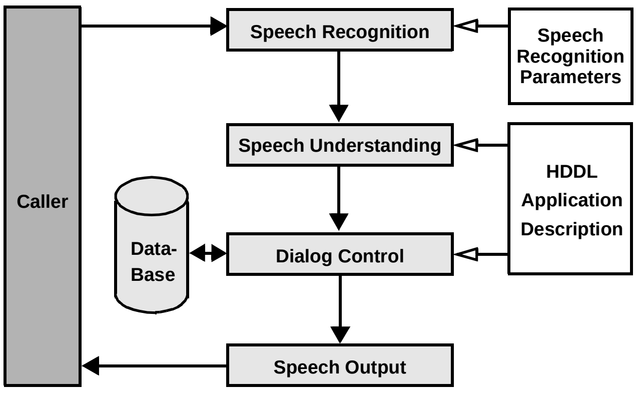

**Figure 1:** The basic architecture of the Philips system.

## 2. THE PHILIPS SYSTEM

Early in 1994, we introduced a train timetable information system that was, as far as we know, the world's first natural-language dialog system available to the general public. This system is described in more detail in [2] and [3]; however, we will sum up its most important ideas here.

### 2.1. System Architecture

The architecture of our system has been based on two main principles: First, it has been our goal right from the beginning not to develop a single optimized application but rather a methodology, along with a development tool, that would allow the creation of applications from various areas, as long as they shared the basic characteristics of an automatic inquiry system. This means that we have separated application-specific details from general mechanisms. Second, we took care to organize the main processing components, namely speech recognition, speech understanding, dialog control, and speech output, as separate, individual modules that are executed sequentially but do not otherwise communicate while a sentence is processed (see Fig. 1). This helped us keep the system simple, efficient, and easily extendable. These components are described briefly in the following sections.

## 2.2. Speech Recognition

The speech recognizer used in our dialog system is based on our standard large-vocabulary continuous-speech HMM recognizer also employed for other purposes, e.g. dictation [9]. However, in our case, the output from the recognizer is not a single best sentence but a *word graph* that contains alternatives. This allows the system to utilize the knowledge contained in the subsequent speech understanding module for the actual computation of the most likely sequence of spoken words.

Initially, the speech recognition component accounted for about 95% of the computational power needed to run the system. While we have been able to improve its efficiency considerably (see section 3.1.), recognition still is the single most time-consuming task.

## 2.3. Speech Understanding

Unlike read speech, spontaneous speech is often grammatically incorrect. In fact, in systems used in the real world, grammatically correct sentences are the exception rather than the rule. In addition to that, the performance of the recognizer in a natural-language dialog system tends to be relatively poor because of the adverse conditions of medium to large vocabulary, speaker independence, telephone-line signal quality, real-time constraint and so on. Word error rates can be as high as 30%.

This means that an attempt to parse a user utterance completely would fail quite often. Therefore, we do not try to do so in our speech understanding component. Instead, we are looking only for those words and word sequences – so-called *concepts* – that carry a meaning with respect to the application. This method has not only shown to yield very good understanding results [1], it is particularly robust towards corrupted speech input. In addition, it is computationally inexpensive.

For the definition of and the search for the concepts, we use an *attributed stochastic context-free grammar* that serves three purposes:

- It is used as a language model;
- it identifies the concepts;
- and it is used to compute their meaning.

## 2.4. Dialog Control

While the sequence of possible questions and actions is relatively constrained in hierarchical, menu-structured dialog systems, the situation is completely different in flexible natural-language systems: In general, almost any question can follow almost any other, depending on user behavior and recognition results. Therefore, using traditional means of describing the dialog flow, e.g. finite state networks, would normally result in specifications of enormous size and complexity, and is generally not possible anymore.

To overcome this problem, we have developed a description language, called HDDL, specifically for the kind of dialogs that are encountered in automatic inquiry systems. The principles of this language and its interpreter were already introduced in [3]; here, we will only briefly recall the basic ideas:

- All information that constitutes the specification of a dialog, in particular phrases to be understood, questions to be asked, and actions to be taken, is organized in so-called *sections* each of which contains a certain class of information.
- For reduced complexity, the information in these sections is usually denoted in a declarative way.
- The HDDL interpreter automatically creates the next question according to a from-specific-to-general strategy, based on the dialog description as well as the actual system status.
- Two *macros* are available for the HDDL programmer that automatically handle questions and confirmation. In particular, they take care of rephrasing in case of repetitions, deal with the processing of special elements in a caller’s response, e.g. expressions for “yes” or “no”, and perform other related tasks.
- A general-purpose transaction interface facilitates the inclusion of arbitrary databases or the interaction with external systems.

## 2.5. Speech Output

Our system contains a very simple language generation component that allows the creation of system questions or other output by concatenating phrases specified in the HDDL program. This process is controlled by specific conditions which in turn have access to the current system status.

While there have been considerable improvements in speech synthesis programs during the last few years, their output still does not sound very natural. Therefore, we have not used any of them for the conversion of the so generated written sentences to spoken ones so far. Instead, we rely on concatenating and replaying pre-recorded phrases.

For a finished system that will be in wide-spread use, it may be well worth the effort to optimize the output created this way. For example, the speech quality can be improved considerably by using prosodically different versions of the same word, depending on the context it appears in, and by cutting the individual phrases out of complete sentences instead of recording them separately and without their appropriate context. The work described in [5] has shown that the resulting system prompts can indeed sound very natural.

## 3. EXTENSIONS AND IMPROVEMENTS

The original ideas, as described in the previous sections, are still the foundation of the current Philips dialog system. Of course, there have been a number of enhancements since its introduction. In the course of the following passages we will outline the most important improvements and some of the new features.

## 3.1. Speeding up the System

Initially, three 275 MHz DEC AXP workstations running in parallel were necessary for coping with the system’s power demands, serving just one telephone line. Two of these machines did the distance calculations for the recognizer; the third one handled all the rest including speech understanding and dialog control. The system has always been a pure software solution, written in C and C++; no specialized hardware is employed.

As already mentioned, the computationally by far most expensive module is the recognizer. Our optimization efforts have therefore been focused on this component. Increasingly efficient algorithms and more sophisticated implementations used here have not only improved the recognition accuracy but have also lead to a substantial reduction of hardware demands.

As a result, we are now able to simultaneously serve 4 lines of a train schedule information system, or a system with comparable vocabulary size and complexity, on a single 266 MHz PC, of course meeting real-time conditions. And fortunately, we can still see possibilities for even further decreased hardware costs.

## 3.2. Feedback from Dialog Control to Speech Understanding

The main data flow follows a sequential path from speech recognition via speech understanding, dialog control, and speech output, as depicted in Fig. 1. For greater flexibility and dialog naturalness, however, it is useful if speech understanding can be modified depending on the current dialog state. Consequently, we introduced a way for adapting this component to specific dialog situations.

These adaptations are specified either implicitly or explicitly in the dialog control part of the HDDL program. For example, in case of confirmations that make use of the macro provided by HDDL, the programmer does not ask for any feedback directly; he or she only writes down the concepts which should be understood in case of a verification. Passing this information to the speech understanding component, and, in particular, enabling or disabling of these concepts is done by the HDDL run-time system automatically and simultaneously with the creation of a system prompt whenever the macro becomes applicable.

For explicit feedback, on the other hand, HDDL statements are available for

- enabling and disabling of specific concepts for the processing of the next utterance,
- defining preferences between concepts, and
- calling or activating complete dialog modules, so-called subdialogs (cf. section 3.6.).

## 3.3. Feedback from Dialog Control to Speech Recognition

A second feedback loop exists that encompasses dialog control and speech recognition: An HDDL statement is available for switching the lexicon, language model, and acoustic model of the recognizer. As a result, the HDDL program can control specialized recognizer configurations for specific dialog situations. Typical examples are the recognition of digits, names, or spelling. Moreover, this feature supports an easy development of multilingual applications.

Feedback information can only be passed inbetween sentences. During the processing of an individual utterance, the data flow follows the usual sequential structure.

## 3.4. Improved Search

The basic version of our dialog system employs word *n*-grams and a stochastic context-free grammar as language models. Because of the way these word-level language models are normally used, they are unable to capture certain rigid or long-range constraints without adequate pre-processing of the training data or manual adaptation of the models. Such constraints, however, can be useful for rejecting inconsistent, and therefore probably misunderstood, user utterances. In an automatic switchboard system, for example, a cooperative caller will never intentionally ask for a non-existent person. Being able to reject an invalid first name / last name combination in the framework of our stochastic language models, however, would either require to include all valid names into the grammar, or to make sure that all correct combinations appear in the training material. For reasons of easy maintenance, neither solution is really viable. In fact, if we look at consistency across several sentences, neither would be able to handle it at all.

Therefore, an improved search in our automatic telephone switchboard system PADIS uses an external database that describes consistent information, e.g. actually existing first and last name combinations, and handles constraints at several levels. In particular, the consistency of

- the user's utterance itself,
- new information vs. the current dialog state, and
- new information vs. the last system prompt

is checked. A flexible *n*-best algorithm is used for computing the next-best sentence from the word graph until consistent information is found. This method reduced the attribute error rate by 27% relatively [8].

## 3.5. Confidence Measures

Dialog systems which make use of confidence measures for recognized semantic items are able to behave more human-like than systems with a rigid verification scheme: The reliability of recognized items can be used to control the extent and manner of verification. For example, understood values with a high reliability will not be verified at all, whereas a particularly poor recognition leads to an explicit confirmation, or even to the rejection of the item.

Experimental results with the PADIS system show that the probabilities delivered by the recognizer can be taken as a direct measure for the reliability of the recognition [7]. In this approach, the probabilities for an *n*-best list are determined and re-normalized for each semantic item of the best sentence. This method is computationally inexpensive, leads to a significant improvement in terms of naturalness of the dialog flow, and reduces the average call duration.

## 3.6. HDDL Subdialogs

The basic ideas and principles of our dialog description language HDDL (cf. section 2.4.) have already been part of the original implementation ([10], [11]). When it came to a redesign of the language, our main emphasis lay on the support of easier application development. All the details of the new syntax and features are described in [12] and [13]; here, we will concentrate on the concept for application modularization that has already been introduced in [4].

HDDL modules are called *subdialogs* and may contain all sections of a complete HDDL program, i.e. an attributed grammar for speech understanding, variable definitions, rules describing consistency and background information, dialog flow specifications, transaction interface, etc. In fact, it is possible to define a complete stand-alone dialog in one single module. Alternatively, an application can be divided into several subdialogs describing specific aspects or useful dialog fragments – very similar to the modularization techniques known from other programming languages.

What sets the subdialog concept apart, however, is a sophisticated calling mechanism that makes it suitable for various situations. A subdialog can either be *called*, *activated* or set to *listen*:

- *Calling* a subdialog is comparable with a function call in programming languages such as C. The current scope is pushed on a stack, parameters are passed and a new scope is entered. This mechanism is appropriate if an application can be divided into largely independent sub-applications with their own speech understanding and dialog control parts.
- If a subdialog is *activated*, its specifications are added to the current scope instead of replacing it, i.e. speech understanding will be based on an extended grammar afterwards, and the dialog flow will be controlled by more actions. The declarative character of HDDL allows the definition of clear semantics for the extended scope. Using activation, even applications with sub-tasks that are strongly related to each other can be modularized. In particular, recurring tasks need only be described once.
- Setting a subdialog to *listen* means adding only its speech understanding part to the current scope. Values for variables of this subdialog can be understood, but accompanying actions will not be performed in this state. A typical example for using the listening mode is an application providing different services, implemented as subdialogs, after an initial selection question:

  *System:* “Do you need price or timetable information?”
  *Caller:* “I need price information for travelling groups.”

  The first part of the caller’s response belongs to the selection question, whereas the second provides already information for the price subdialog. By setting both the price and the timetable subdialogs to listen, the complete answer can be processed. Subsequently, the desired sub-application can be called without losing the information already given by the user.

Fig. 2 shows the current architecture of our system that incorporates the modifications described above.

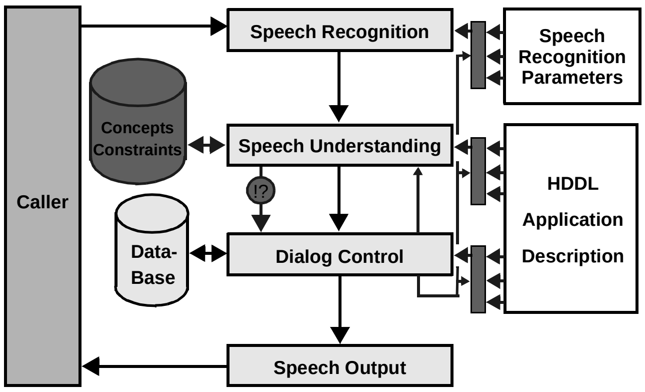

**Figure 2:** The improved architecture of the Philips system. While we retained the basic principles, in particular the sequential execution of the processing modules, it is now much more easily possible to adapt the system’s behavior to individual dialog situations, controlled by the HDDL application description and the actual system status: Concepts and constraints can be specified on an external database, the understanding module provides confidence measures to affect the dialog flow, and feedback from dialog control can be used to modify the behavior of speech recognition, speech understanding, and dialog control itself, for example by making use of HDDL subdialogs.

## 4. IMPLEMENTED APPLICATIONS

Our system has been used for the realization of many applications from a variety of areas, in different languages, complex or simple, most notably the ones listed below. All status information given is as of February 1998.

- Probably best known is our already mentioned train timetable inquiry system in the German language, called TABA for “Telefonische Automatische Bahnfahrplan-Auskunft” (telephone-based automatic train timetable information), which has been publicly available under +49-241-604020 since February 1994. This system can provide information on connections between about 1,000 German cities. It incorporates a database that contains real connection data, so people have been using it from the very beginning if they wanted to obtain an actual connection, providing us with real-life speech data.

  This system is not related to the information service of Deutsche Bahn, the main operator of the German railway system, which means in particular that it is neither advertised nor otherwise commercially supported. Nonetheless, demand has been so high that we added a second telephone line in 1995. On average, we receive about 2,500 calls per month, with a success rate[1] of more than 90%.

  For a detailed description of the system and our experience gained from its operation, see [2].
- A similar system, also in German, has been implemented for Swiss Railways (SBB), and has been operational since June 1996. The system handles about 3,000 railway stations and bus stops, many of which do not have German but French or Italian names. It serves four telephone lines (additional lines are currently being installed) and is part of SBB’s comprehensive travel information service. Human-operated schedule information is still available under a separate telephone number, but at a higher cost for the caller.

  SBB did an extensive survey on this system in 1996 [6], based

1. The success rate, our standard quality measure for inquiry systems, is the percentage of appropriate calls that were completed successfully, i.e. in the course of which the caller actually got the information he or she apparently wanted. “Appropriate calls” are calls that the system by design can handle; if a caller tries to, say, order a pizza from a train schedule information system, the call is not counted as a failure but is simply ignored.

on a questionnaire appended to the system itself, as well as direct customer contacts. It turned out that 93% of the calls that fell into the system's domain were completed successfully; in addition, more than 80% of the callers asked expressed satisfaction and will use the system again. These figures show clearly that natural-language dialog systems can indeed be operated highly successfully under real-world conditions.

- A Dutch travel information system, serving 15 lines, has become available early in 1998; a similar system in French language is currently under development.
- A system that provides information whether a flight is on time was introduced by Lufthansa airways in June 1997. Its language is German; however, it allows the speaking of foreign city names in their respective language (English, French, Spanish etc.). Since the recognizer is based on German acoustics, the foreign names were specifically transcribed using the German phoneme set, resulting in somewhat strange but effective transcriptions.
  This system was installed as part of an already existing menu-structured environment and serves 12 lines. At a point in the menu hierarchy at which callers had been referred to human operators before, they are now connected to our system. Wait times are down, and the success rate is better than 90%.
- The same basic multilingual approach is employed in a weather information system that is currently in the testing phase. It has been implemented by Deutsche Telekom, using our dialog system development toolkit. For full roll-out, 8 locations with 60 lines each are planned.
- A restaurant guide for the Boston area, in American English, is our American demonstration system. It is available under +1-770-350-6212. A caller can specify the desired type of food (e.g. Italian or French), the preferred location, and a price range. Similar systems exist for Atlanta and Toronto, the latter one currently being extended to a comprehensive yellow pages service. This project is done together with a major Canadian telecommunications company.
- *SpeechAttendant* is, as the name implies, a generic automatic telephone attendant installed at several sites in Germany. Callers can ask, as usual in fluent speech, for information on up to 250 employees of a company, e.g. telephone or fax number, office location, e-mail address etc., and can be connected to a desired extension. A more basic version has been widely used within Philips for more than two years, and has rendered traditional telephone lists largely unused.

## 5. CONCLUSION AND FURTHER WORK

As we could learn by developing, installing, and operating the applications briefly described above, our approach to natural-language dialog systems is well suited for the various areas and problems addressed, not only in the laboratory, but also in the real world and with real everyday users. In fact, while there is clearly room for improvements of our core technology itself, it appears nonetheless that we have now reached a point at which it is more important to address the various aspects of the environments our systems are used in.

A typical example is the audio/telephony interface, for which a number of topics will have to be looked into: Standard telephony features like barge-in or DTMF recognition will have to be supported. When more than just a few lines need to be installed for a service, a scalable client/server architecture will be needed. Applications on the client side will have to communicate across a LAN with the server that provides the telephone network access.

On the other hand, it is very important, and equally challenging, to speed up the application creation process. The first step has been the introduction of the HDDL modularization concept. A next step should be the development of re-usable HDDL subdialogs and pertaining language models, a task not as simple and straightforward as it may appear at first glance. A comfortable, easy-to-use HDDL development environment, as well as the improvement of already existing tools, is just as necessary. It should be noted, however, that even today the collection of adequate training material for the acoustic model and the language models is clearly more time-consuming than the actual application development itself.

## REFERENCES

1. H. Aust, M. Oerder: "Database Query Generation from Spoken Sentences". 2nd Workshop on Interactive Voice Technology for Telecommunications Applications (IVTTA 94), Kyoto, Japan, pp. 141–144, Sept. 1994.
2. H. Aust et al.: "The Philips Automatic Train Timetable Information System". Speech Communication 17, pp. 249–262, Nov. 1995.
3. H. Aust, M. Oerder: "Dialogue Control in Automatic Inquiry Systems". ESCA Workshop on Spoken Dialogue Systems, Aalborg, Denmark, pp. 121–124, June 1995. Also published in: 9th Twente Workshop on Language Technology, Enschede, The Netherlands, pp. 45–49, June 1995.
4. H. Aust, N. Lenke: "The Philips Dialog System – Applications and Improvements". COST Telecom Workshop, Rhodes, Greece, pp. 25–32, Sept. 1997.
5. E.A.M. Klabbers: "High-Quality Speech Output Generation through Advanced Phrase Concatenation". COST Telecom Workshop, Rhodes, Greece, pp. 85–88, Sept. 1997.
6. J.-C. Peng, R. Flückiger, S. Häfeli: "Prisma Voice Konzept". SBB-internal document, Oct. 1996.
7. B. Rueber: "Obtaining Confidence Measures from Sentence Probabilities". 5th European Conf. on Speech Communication and Technology (EUROSPEECH 97), Rhodes, Greece, pp. 739–742, Sept. 1997.
8. F. Seide, B. Rueber, A. Kellner: "Improving Speech Understanding by Incorporating Database Constraints and Dialogue History". 4th Int. Conf. on Spoken Language Processing (ICSLP 96), Philadelphia, PA, USA, pp. 1017–1020, Oct. 1996.
9. V. Steinbiss et al.: "Continuous-Speech Dictation – From Theory to Practice". Speech Communication 17, pp. 19–38, Aug. 1995.
10. "SpeechMania 1.0: HDDL Reference Manual". Philips Dialogue Systems, Aachen, Germany, 1996.
11. "SpeechMania 1.0: HDDL User's Guide". Philips Dialogue Systems, Aachen, Germany, 1996.
12. "SpeechMania 2.0: HDDL Reference Manual". Philips Dialogue Systems, Aachen, Germany, 1997.
13. "SpeechMania 2.0: HDDL User's Guide". Philips Dialogue Systems, Aachen, Germany, 1997.